

CS105

Introduction to Object-Oriented Programming

Prof. Dr. Nizamettin AYDIN

naydin@itu.edu.tr

nizamettin.aydin@ozyegin.edu.tr

Constructors And Method Overloading

Outline

- Bank Account – version 3
- Constructors
- Calling constructors
- No constructors
- Bank Account – version 4
- Multiple Constructors
- Bank Account – version 5
- Bank Account – version 6
- Overloading Functions
- Bank Account – version 7

Bank Account – version 3

```
public class AccountTest {  
    public static void main(String[] args) {  
        Account account1 = new Account();  
        account1.number = 1;  
        account1.balance = 100;  
        account1.currency = "TL";  
  
        Account account2 = new Account();  
        account2.number = 2;  
        account2.balance = 200;  
        account2.currency = "USD";  
  
        account1.report();  
        account2.report();  
  
        // Deposit 50 TL into account 1  
        account1.deposit(50);  
  
        // Deposit 300 USD into account 2  
        account2.deposit(300);  
  
        account1.report();  
        account2.report();  
    }  
}
```

Constructors

- block of codes which are automatically called when we create objects (when an instance of the class is created).
 - a special type of method which is used to initialize the object.
 - Every time an object is created using the new() keyword, at least one constructor is called.

```
// Constructor
public Account(int n, double b, String c) {
    number = n;
    balance = b;
    currency = c;
}
```

- It looks like other methods, but...
 - It has the same name with the class
 - It does not have a return type,
 - but it actually returns the reference to (address of) the constructed object

Calling constructors

```
Account account1 = new Account();  
account1.number = 1;  
account1.balance = 100;  
account1.currency = "TL";
```

```
Account account1 = new Account();
```

```
// Constructor  
public Account(int n, double b, String c) {  
    number = n;  
    balance = b;  
    currency = c;  
}
```

No constructors

```
Account account1 = new Account();  
account1.number = 1;  
account1.balance = 100;  
account1.currency = "TL";
```

- We did not have any constructors before.
- How did we create objects without the constructor?
 - If there is no explicit constructor, then the default constructor is used.
 - Default constructors do not take any parameters.

Bank Account – version 4

```
public class Account {
    int number;
    double balance;
    String currency;

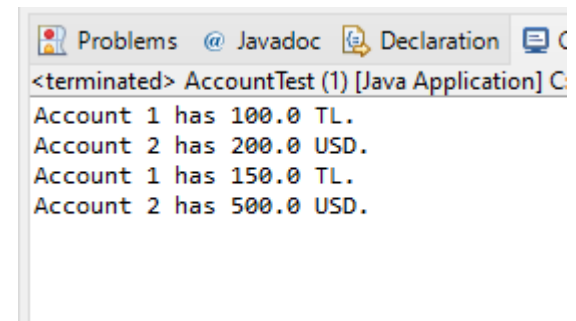
    // Constructor
    public Account(int n, double b, String c) {
        number = n;
        balance = b;
        currency = c;
    }

    public void deposit(double d) {
        balance = balance + d;
    }

    public void report() {
        System.out.println("Account " + number
            + " has " + balance
            + " " + currency + ".");
    }
}
```


Bank Account – version 4

```
public class AccountTest {  
    public static void main(String[] args) {  
        Account account1 = new Account(1, 100, "TL");  
  
        Account account2 = new Account(2, 200, "USD");  
  
        account1.report();  
        account2.report();  
  
        // Deposit 50 TL into account 1  
        account1.deposit(50);  
  
        // Deposit 300 USD into account 2  
        account2.deposit(300);  
  
        account1.report();  
        account2.report();  
    }  
}
```



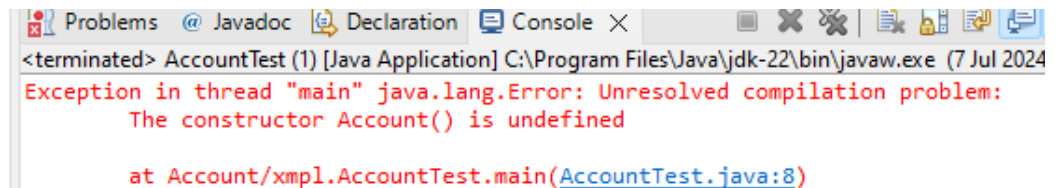
The screenshot shows a console window with the following output:

```
<terminated> AccountTest (1) [Java Application] C  
Account 1 has 100.0 TL.  
Account 2 has 200.0 USD.  
Account 1 has 150.0 TL.  
Account 2 has 500.0 USD.
```

Why do we get the following error?

```
public class AccountTest {  
    public static void main(String[] args) {  
        Account account1 = new Account(1, 100, "TL");  
  
        Account account2 = new Account();  
        account2.number = 2;  
        account2.balance = 200;  
        account2.currency = "USD";  
  
        account1.report();  
        account2.report();  
  
        // Deposit 50 TL into account 1  
        account1.deposit(50);  
  
        // Deposit 300 USD into account 2  
        account2.deposit(300);  
  
        account1.report();  
        account2.report();  
    }  
}
```

When a constructor (with parameters) is implemented, then the system does not provide a default (without parameters) constructor.



```
Problems @ Javadoc Declaration Console X  
<terminated> AccountTest (1) [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (7 Jul 2024  
Exception in thread "main" java.lang.Error: Unresolved compilation problem:  
    The constructor Account() is undefined  
  
    at Account/xmpl.AccountTest.main(AccountTest.java:8)
```

Default Constructor

- When a constructor (with parameters) is implemented, then the system does not provide a default (without parameters) constructor.
 - Can we implement our own constructor without parameters?
 - **Yes, we can...**
 - A class can have **multiple constructors**.
 - This is possible by overloading constructors.
- **Method overloading** gives us the capability to implement a particular function in different ways.
 - Overloaded functions will have the same name but different function arguments.

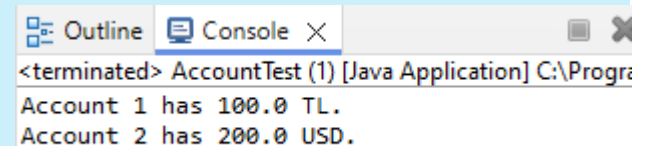
Multiple Constructors

```
// Constructors
public Account() {

}
public Account(int n, double b, String c) {
    number = n;
    balance = b;
    currency = c;
}
```

```
public class AccountTest {
    public static void main(String[] args) {
        Account account1 = new Account(1, 100, "TL");

        Account account2 = new Account();
        account2.number = 2;
        account2.balance = 200;
        account2.currency = "USD";
    }
}
```



Outline Console X
<terminated> AccountTest (1) [Java Application] C:\Progr
Account 1 has 100.0 TL.
Account 2 has 200.0 USD.

Multiple Constructors

- Can we have more than two overloaded constructors?
- **Yes we can...**

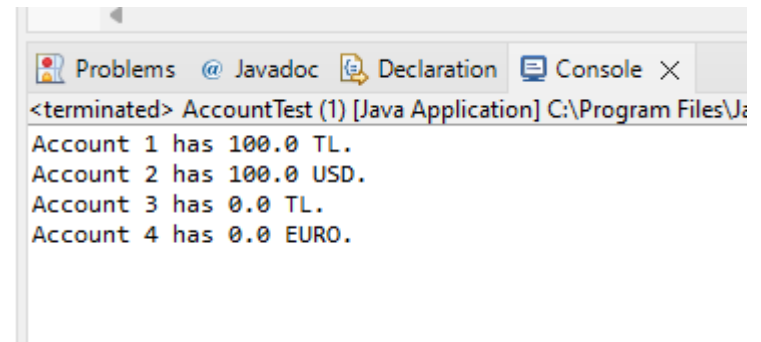
```
// Constructors
public Account() {

}
public Account(int n, double b, String c) {
    number = n;
    balance = b;
    currency = c;
}
public Account(int n, String c) {
    number = n;
    balance = 0;
    currency = c;
}
public Account(int n) {
    number = n;
    balance = 0;
    currency = "TL";
}
```

All these constructors do the same thing which is creating an object, but what they assign to the class instances are different.

Bank Account – version 5

```
public class AccountTest {  
    public static void main(String[] args) {  
        Account account1 = new Account(1, 100, "TL");  
  
        Account account2 = new Account();  
        account2.number = 2;  
        account2.balance = 200;  
        account2.currency = "USD";  
  
        Account account3 = new Account(3);  
  
        Account account4 = new Account(4, "EURO");  
  
        account1.report();  
        account2.report();  
        account3.report();  
        account4.report();  
    }  
}
```



The screenshot shows a console window with the following output:

```
<terminated> AccountTest (1) [Java Application] C:\Program Files\J  
Account 1 has 100.0 TL.  
Account 2 has 100.0 USD.  
Account 3 has 0.0 TL.  
Account 4 has 0.0 EURO.
```

Lets add more to our account!

- interest rate (double)

```
int number;  
double balance;  
String currency;  
double interestRate;
```

Modify the constructors

```
// Constructors
public Account() {

}
public Account(int n, double b, String c, double i){
    number = n;
    balance = b;
    currency = c;
    interestRate = i;
}
public Account(int n, String c){
    number = n;
    balance = 0;
    currency = c;
    interestRate = 0;
}
public Account(int n){
    number = n;
    balance = 0;
    currency = "TL";
    interestRate = 0;
}
```


Add more constructors

```
public Account (int n, double b, String c){  
    number = n;  
    balance = b;  
    currency = c;  
    interestRate = 0;  
}
```

```
Account account5 = new Account(5, 200, "TL");
```

```
public Account (int n, double i, String c){  
    number = n;  
    balance = 0;  
    currency = c;  
    interestRate = i;  
}
```

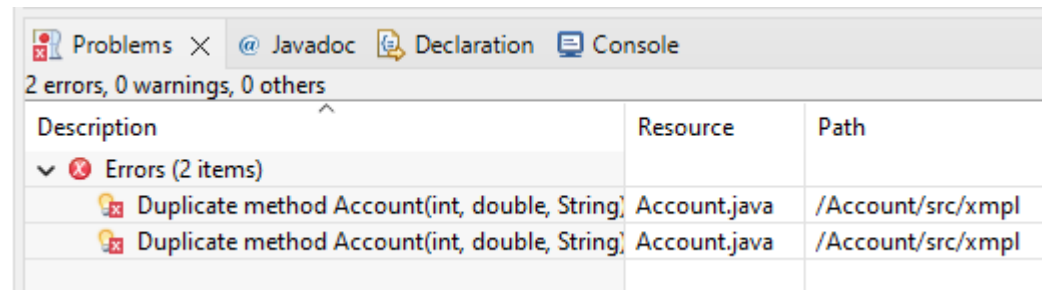
```
Account account6 = new Account(6, 0.02, "TL");
```

- Do you see any problem?

Add more constructors

- You can have multiple constructors as long as they have different argument lists.

```
public Account (int n, double b, String c) {  
    number = n;  
    balance = b;  
    currency = c;  
    interestRate = 0;  
}  
public Account (int n, double i, String c) {  
    number = n;  
    balance = 0;  
    currency = c;  
    interestRate = i;  
}
```



- System differentiates constructors based on their argument lists, therefore two constructors with same argument list cause compiler error.

–Duplicate method error!

Any idea to fix this?

Multiple Constructors

```
public Account (int n, double b, String c) {  
    number = n;  
    balance = b;  
    currency = c;  
    interestRate = 0;  
}  
public Account (int n, String c, double i) {  
    number = n;  
    balance = 0;  
    currency = c;  
    interestRate = i;  
}
```

- Same type of arguments, but their order is different!

Bank Account – version 6

- Be careful when calling these functions!

```
Account account1 = new Account(1, 100, "TL");
```

```
Account account2 = new Account();
```

```
Account account3 = new Account(3);
```

```
Account account4 = new Account(4, "EURO");
```

```
Account account5 = new Account(5, 200, "TL");
```

```
Account account6 = new Account(5, "TL", 0.02);
```

Bank Account – version 6

```
public Account() {  
    }  
public Account(int n, double b,  
    number = n;  
    balance = b;  
    currency = c;  
    interestRate = i;  
}  
public Account(int n, String c){  
    number = n;  
    balance = 0;  
    currency = c;  
    interestRate = 0;  
}  
public Account(int n){  
    number = n;  
    balance = 0;  
    currency = "TL";  
    interestRate = 0;  
}
```

```
public Account(int n, double b, String c){  
    number = n;  
    balance = b;  
    currency = c;  
    interestRate = 0;  
}  
public Account(int n, String c, double i){  
    number = n;  
    balance = 0;  
    currency = c;  
    interestRate = i;  
}
```

```
Account account1 = new Account(1, 100, "TL");  
Account account2 = new Account();  
Account account3 = new Account(3);  
Account account4 = new Account(4, "EURO");  
Account account5 = new Account(5, 200, "TL");  
Account account6 = new Account(5, "TL", 0.02);
```

Overloading Functions

- We have overloaded the constructor.
- Can we overload other methods as well?
- **Yes, we can...**
-
- **Overloading deposit function**

```
public void deposit(double d) {  
    balance = balance + d;  
}  
public void deposit() {  
    balance = balance + 0;  
}
```

Bank Account – version 7

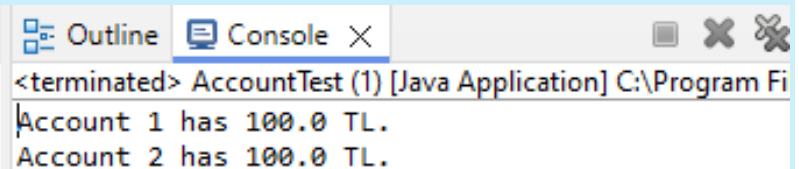
```
public void deposit(double d) {
    balance = balance + d;
}

public void deposit() {
    balance = balance + 0;
}
```

```
public static void main(String[] args) {
    Account account1 = new Account(1, 100, "TL");
    Account account2 = new Account(2);

    account2.deposit(100);
    account1.deposit();

    account1.report();
    account2.report();
}
```



The screenshot shows a console window titled "<terminated> AccountTest (1) [Java Application] C:\Program Fi". The output text is:

```
Account 1 has 100.0 TL.
Account 2 has 100.0 TL.
```

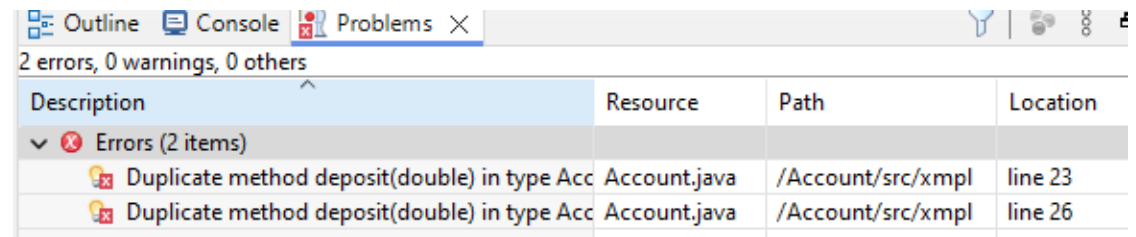
Overloading deposit method

- In addition to our two deposit methods, can we have the following method as well?

```
public double deposit(double m) {  
    balance = balance + m;  
    return balance;  
}
```


Overloading deposit method

```
public double deposit(double d) {  
    balance = balance + d;  
}  
public double deposit(double m) {  
    balance = balance + m;  
    return balance;  
}  
public double deposit() {  
    balance = balance + 0;  
}
```



The screenshot shows the 'Problems' window in an IDE. It displays two error messages: 'Duplicate method deposit(double) in type Acc' at line 23 and 'Duplicate method deposit(double) in type Acc' at line 26. The table below summarizes the error details.

Description	Resource	Path	Location
✖ Duplicate method deposit(double) in type Acc	Account.java	/Account/src/xmpl	line 23
✖ Duplicate method deposit(double) in type Acc	Account.java	/Account/src/xmpl	line 26

- Overloaded methods need to have different function arguments (parameter list)
 - If the arguments are same but the return type is different, we will still get compiler error

www.javatpoint.com/method-overloading-in-java

- Why Method Overloading is not possible by changing the return type of method only?
 - In java, method overloading is not possible by changing the return type of the method only because of ambiguity.
 - Let's see how ambiguity may occur:

```
class Adder{
    static int add(int a,int b){return a+b;}
    static double add(int a,int b){return a+b;}
}

class TestOverloading3{
    public static void main(String[] args){
        System.out.println(Adder.add(11,11)); //ambiguity
    }
}
```

Compile by: javac TestOverloading3.java

- Output

```
Overloading3.java:3: error: method add(int,int) is already defined in class Adder
static double add(int a,int b){return a+b;}
    ^
1 error
```

www.javatpoint.com/method-overloading-in-java

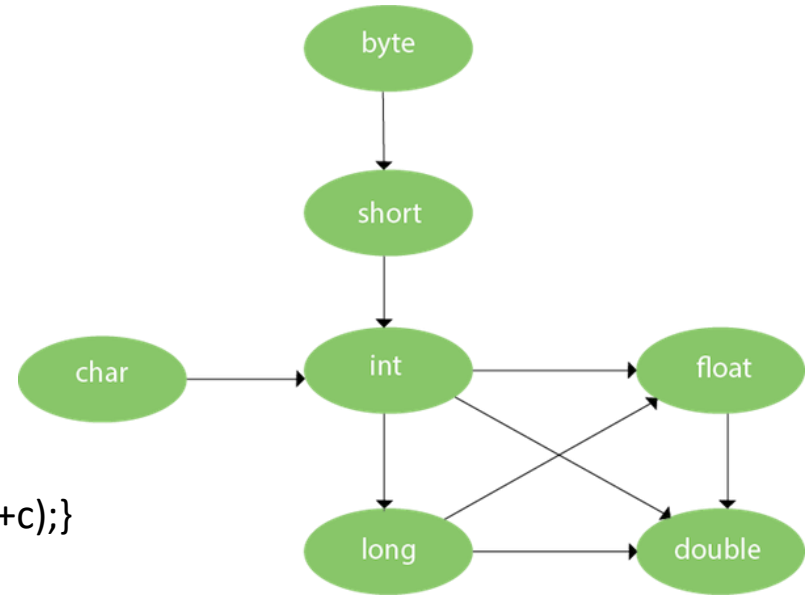
- Method Overloading and Type Promotion:

- One type is promoted to another implicitly if no matching datatype is found.

- As displayed in the diagram, byte can be promoted to short, int, long, float or double.
- The short datatype can be promoted to int, long, float or double.
- The char datatype can be promoted to int, long, float or double and so on.

Example of Method Overloading with TypePromotion:

```
class OverloadingCalculation1{  
    void sum(int a,long b){System.out.println(a+b);}  
    void sum(int a,int b,int c){System.out.println(a+b+c);}  
  
    public static void main(String args[]){  
        OverloadingCalculation1 obj=new OverloadingCalculation1();  
        obj.sum(20,20); //now second int literal will be promoted to long  
        obj.sum(20,20,20);  
    }  
}
```



Compile by: javac OverloadingCalculation1.java

Run by: java OverloadingCalculation1

```
40  
60
```

www.javatpoint.com/method-overloading-in-java

- Example of Method Overloading with Type Promotion if matching found:

–If there are matching type arguments in the method, type promotion is not performed.

```
class OverloadingCalculation2{  
    void sum(int a,int b){System.out.println("int arg method invoked");}  
    void sum(long a,long b){System.out.println("long arg method invoked");}  
  
    public static void main(String args[]){  
        OverloadingCalculation2 obj=new OverloadingCalculation2();  
        obj.sum(20,20);//now int arg sum() method gets invoked  
    }  
}
```

Compile by: javac OverloadingCalculation2.java

Run by: java OverloadingCalculation2

- Output:

```
int arg method invoked
```

www.javatpoint.com/method-overloading-in-java

- Example of Method Overloading with Type Promotion in case of ambiguity:
 - If there are no matching type arguments in the method, and each method promotes similar number of arguments, there will be ambiguity.

```
class OverloadingCalculation3{  
    void sum(int a,long b){System.out.println("a method invoked");}  
    void sum(long a,int b){System.out.println("b method invoked");}  
  
    public static void main(String args[]){  
        OverloadingCalculation3 obj=new OverloadingCalculation3();  
        obj.sum(20,20);//now ambiguity  
    }  
}
```

```
Compile by: javac OverloadingCalculation3.java  
  
gCalculation3.java:7: error: reference to sum is ambiguous  
obj.sum(20,20);//now ambiguity  
    ^  
d sum(long,int) in OverloadingCalculation3 match  
1 error
```

- Output:

Any Questions?